University of Illinois
IT Professionals Forum
at Urbana - Champaign

ELI Engineering's Linux Management Environment

IT EXTREME

Spring 2017

Timeline

- “The Surge”
  - Started Jan 2014, Targeted End: Jan 2015
  - Real End: Summer 2015 (1.5 years)
  - Standardized on CFEngine, Cobbler, SL6
- “TheLinux 2.0”
  - What we called ELI before we had a name
  - Start: July 31, Summer 2015
  - End: Start of Fall Semester 2016

Goals and Objectives

Lifecycle and Integration

Goals/Objectives
- **To Develop tools and processes that apply across Linux generations**
- **To coexist with org needs and best practices**

Notes
- Inventory/End-of-life metadata
- Dinosaur Control! - Sys Age mgmt
- Continuous Transition Process
- Central logging
- More modern infrastructure/tools - git, cfengine, cobbler
- Make sense within greater org policies (AD Audit)
- Upgrade path for SL6
- Backwards compatibility
- Client builds not dependent on backend/infra builds
- Easy to roll out, duplicate many systems

Data Access

Goals/Objectives
- **To provide access to data on modern storage in a secure and flexible manner**

Notes
- Home fileserver compat w/ newer NFS
- File shares with user accessible snapshots
- Integration with user cloud storage
- Enhanced security on homes/shares

Campus Integration

Goals/Objectives
- **To integrate our linux deployments with campus services to provide an easy and intuitive method for users to access, use and share systems, services, and data**

Notes
- Other campus services available to clients (web stuff, box, campus cluster, etc)
- Reuse existing resources when possible
- AD integration - authentication, permisssions
- Common authentication & authorization
- Leverage campus authoritative authentication/authorization + general services

Processes and Documentation

Goals/Objectives
- **To produce salient, reproducible, and consistent practices internally**

Notes
- Standards for scripts/tools created here
- A full featured test environment and processes
- Include as few in-house tools as possible
- Documentation and decision log for core infra
- Granular software deployment
- Proper and full dev environment
- Sane licensing of software controls
- Self-documenting
- Process for component and service requests
- Understandability and documentation
- Technical support from manufacturer or developers
- Provision for "islands"

Pain Points

Goals/Objectives
- **To attempt to reduce limitation of the current managed linux environment**

Notes
- Larger var part
- Install on system under 20 GB
- Symlink controls (~~www -> /home~~)
- Network + Locally Deployable
- Cluster support
- Works on the cloud
- Discretionary access control for admins
- Deals w/ H.W. acceleration dependent window managers
- Granular Security (Frank)
- Control system updates

For the User

Goals/Objectives
- **To provide easy, diverse, and flexible user solutions**

Notes
- "Some" level of support for BYO machines/devices
- Self provisioning of systems
- Encapsulation and isolation of environments
- Makes Linux meet client needs without making IT crazy
- Give the user control/choices
- Multiple window managers - Cinnamon/Mate/GnomeShell
- Helpldesk role in Linux support, management
- Flexible to meet customer needs/desires
- Printing just works
- Software versioning (Matlab 2014/2015/2016/…)
- Mainstream stable OS selection
- Handle kernel upgrades
- Distro agnostic
- Flexible in SW and config methods applied
- Because they want Ubuntu
- Low touch baseline

Flexible, Modular, Highly Available

Goals/Objectives
- **To provide robust and customizable solutions**

Notes
- Independent/modular pieces - PXE install, policy updates, etc
- Options for local replication in case of network failure
- High availability & no single point of failure
- Mobile capable
- Handle disconnected systems
- Leverage existing features - cobbler, software, policies
- Cobbler, ipmi support, ipam, - Do Cobbler better
- Enterprise Container Management
- Keep module like system
- Distributed module sources - like Local@ARI

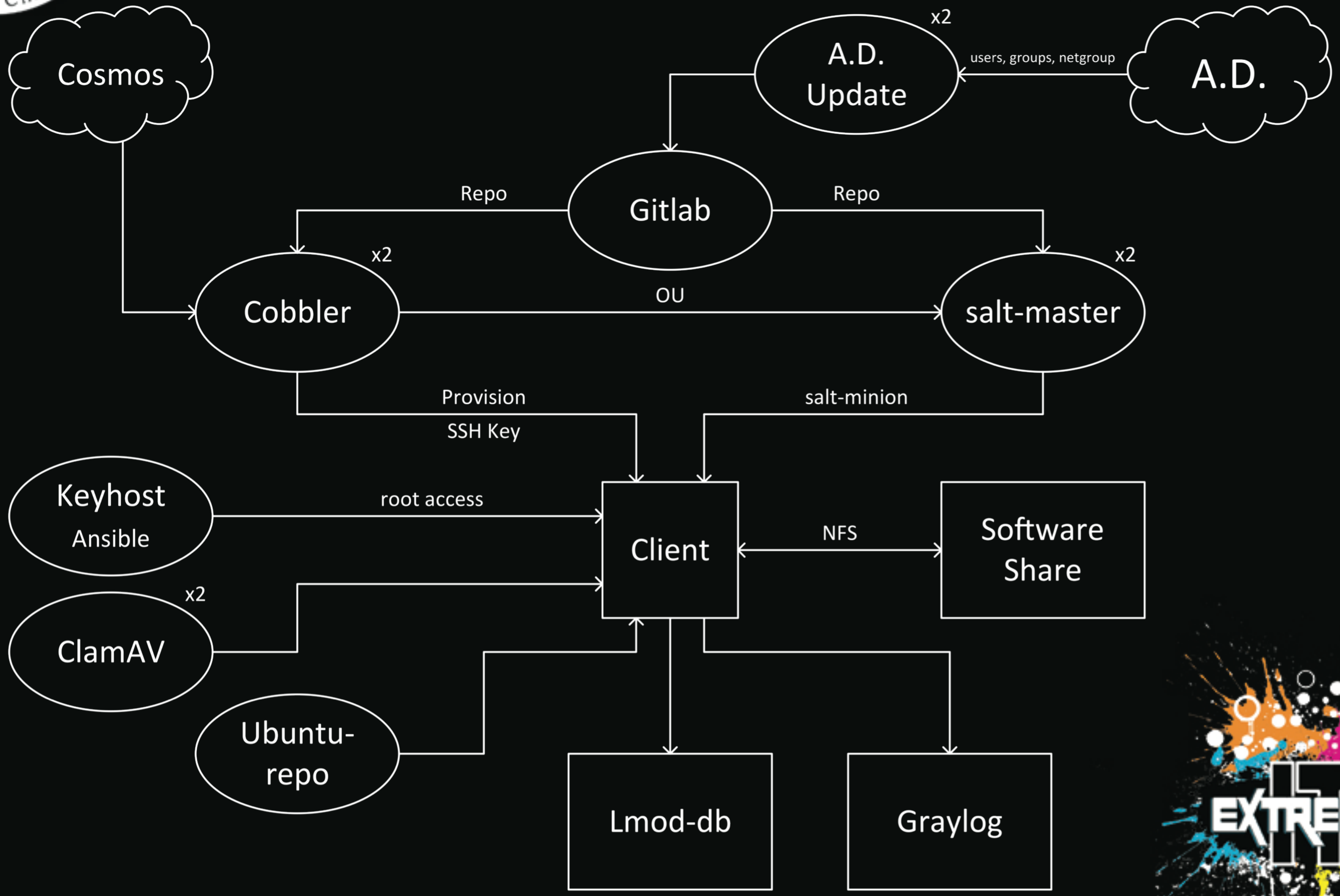
Components

- Provisioning/OS Deployment
- Systems Database/Inventory
- Configuration Management
- Software/Package Management
- Authn/Authz
- File sharing
- Lifecycle Management

Integration

- Sprints:
  - Integration 1A Start Date: October 9, 2015
  - Integration 1B Start Date: November 2, 2015
  - Integration 2A Start Date: January 4, 2016
  - Integration 2B Start Date: February 1, 2016
  - Integration 3A Start Date: February 8, 2016
  - Integration 3B Start Date: February 29, 2016
  - Integration 4A Start Date: March 21, 2016
  - Integration 4B Start Date: April 24 2016
- Component Level Testing
  - Verify that components can interact successfully.
- Solution Level Testing
  - Combination of components can provide the needful

ELI Infra Diagram



Spring 2017

Key Differences

- Flexibility
  - One size fits all vs. meets individualized needs
- Modularity
  - Monolothic design vs. Component design
  - TheLinux all-or-nothing vs. ELI pick and choose
- Highly Available
  - Single point of failure vs. no single points of failure

ELI Provisioning

Provisioning

- What were we looking for?
  - Baremetal provisioning
  - Supports RedHat/CentOS
  - Supports Debian/Ubuntu
- What products did we consider?
  - Cobbler
  - Foreman
  - Satellite/Spacewalk
  - JuJu

Provisioning (cont.)

- ~~JuJu~~
  - Does not support RedHat/CentOS
- ~~Foreman~~
  - Requires Puppet just to install
  - Assumes you will use Puppet as config management
- Satellite/Spacewalk
  - Uses Cobbler under the hood
  - Satellite was $$$$
  - Spacewalk had uncertain future due to release of Satellite 6
- Winner is:
  - Cobbler

Systems Database

- What we wanted:
  - Store the following:
    - Machine Name
    - Machine Model
    - Machine Serial Number
    - Operating System Distribution version
    - Machine Owner
    - OU
    - Warranty End Date
    - Location
    - Machine Birthdate
  - Integration with Cobbler

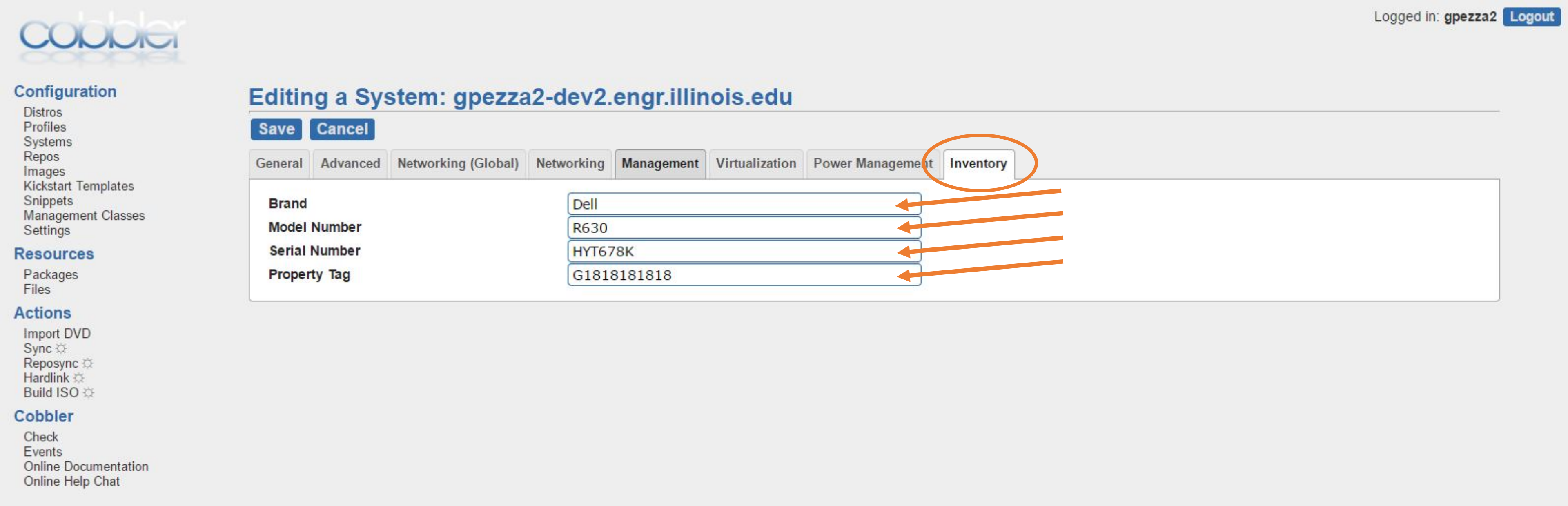
Systems Database (cont.)

- What products did we consider?
  - OCS Inventory
  - Cobbler
  - Tech Services CDB
  - AITS CMDB
  - DIY
- DIY was last option
- Tech Services CDB
  - Too simplistic
  - Not extensible
- OCS Inventory
  - Too complex
  - Required agent
- AITS CMDB
  - Did not have REST API ready for others
  - Needed for Cobbler integration
- Cobbler wins again!

Systems Database (cont.)

- How does one use a provisioning tool as a systems database?
  - The same way you mold steel…heat the hell out of it and bang it with a hammer!
- Cobbler keeps a database (JSON) of all systems
- Made sense to see if we could just add some more metadata fields
- Written in Python with Django web frontend.
- Find the right files, and edit the source code.

Screenshots



Editing a System: gpezza2-dev2.engr.illinois.edu

Brand	Dell
Model Number	R630
Serial Number	HYT678K
Property Tag	G1818181818

University of Illinois IT Professionals Forum at Urbana - Champaign

ELI Configuration Management

IT EXTREME

Spring 2017

So many choices

There are many options when choosing a configuration management system:

- Ansible
- SaltStack
- Puppet
- Chef
- CFEngine
- Fabric
- etc...

Our requirements

Our requirements for a config management system were:

- Easy to install, configure, and automate
- Modular
- Doesn't require special software compilation
- Doesn't need a special SDK
- Doesn't have crazy dependencies
- Supports running from a Git checkout
- Is idempotent - (only changes configs when it needs to)
- Is at least somewhat self-documenting
- Isn't overly complicated and doesn't have too many moving parts
- Continuous management - (not fire-and-forget)

The finalists

After testing and deliberating for a few months on which system to use, we had two finalists:

- Ansible
- SaltStack

Why Ansible?

Ansible is the new hot thing in the world of config management

- Very simple to use
- Agentless - (Doesn't require a special client to be installed on the system)
- Reasonably self-documenting
- Very small and modular
- The “master” server can be anything - (laptop, VM, physical server, etc.)
- Written in Python
- Supports acting on external data from things like cobbler
- Officially supported and backed by RedHat

Why SaltStack?

SaltStack is like a better version of Puppet written in Python instead of Ruby:

- Still pretty simple to use
- Reasonably self-documenting
- Modular
- Written in Python
- Supports acting on external data from things like cobbler

It's a traditional client/server setup where an agent is required on the systems being managed and uses a special key to authenticate the client to the master.

First we tried Ansible

- Ansible seemed like the thing to try if we wanted to be forward thinking and modular.
- Plus it's easy to use and new admins could get up to speed quickly.

Scaling Issues

- We pushed our configs to 400+ freshly built hosts
- It took over 45 minutes to push our configs to these 400 hosts and some of them broke in the process.
- 190 of the 417 were left in a completely unusable and inaccessible state
- Ansible is insanely CPU intensive
- We were seeing file descriptor out of range errors during ansible runs before they failed
- Yum seems to get corrupted very easily from failed ansible runs
- The number of forks can be an issue

Ansible tuning

For several weeks, we tried to eke more performance out of Ansible.

- We configured it the master to use Redis for fact caching so that it wouldn't have to gather all the facts on every run.
- We tweaked SSH settings and enabled pipelining
- We re-wrote roles and added/removed dependencies

Ansible pull

There is an option to use Ansible in a “client/server” manner similar to traditional config management systems.

Ansible pull performs a git pull from a git repo and runs the configs locally on the machine.

This makes things go way faster, but at this point, we’re basically just re-implementing a client/server system. Also you have to be very careful when doing this since there is no reporting in this system and all clients are pulling from the same git repo.

Time was running out

The start of the fall semester was approaching and we weren’t comfortable attempting to use Ansible for our new systems because of our problems with scaling.

We went with SaltStack

In a little over a week, we ported all of our configs that we had written in Ansible to SaltStack and had a running system that scaled to the amount of machines that we needed to manage.

Best features

- We're able to assign clients to specific environments using cobbler metadata, then act on those clients with SaltStack
- We have separate environments for things like:
  - Infrastructure
  - EWS
  - CBTF
  - CS VM Farm
  - Research
  - Dev/Test
- Pillar data
  - Can make use of "sensitive" data in configs without transferring the data to the clients (like Ansible's vault)
- A la carte
  - We can pick and choose exactly which states or "roles" we want to apply in each environment

We’re releasing our configs

We have made a public version of our SaltStack configs available for all.

Hosted on our Gitlab server:

https://gitlab.engr.illinois.edu/engrit-public

Software

Spring 2017

What we wanted

Lifecycle and Integration

- Inventory/End-of-life metadata
  - Multiple software versions
  - When to deploy new versions
  - When to retire old versions
- Central logging
  - Software usage statistics
  - Software versions most commonly used
  - Concurrency of usage
  - “Hot spots” in instructional labs or research groups
- Backwards compatibility
  - New software installations should be available to old clients as well as new
- Client builds not dependent on backend/infra builds
  - Software to be available for all offered client builds

Processes and Documentation

- Granular software deployment
- Sane licensing of software controls
- Self-documenting
  - Software names clear and apparent to users
- Process for component and service requests
  - should have an easy way to request new or updated client software or modules
- Provision for "islands"
  - NDA considerations
  - Licensing restrictions

For the User

- "Some" level of support for BYO machines/devices
  - Documentation or one-click installers for Citrix / remote workstations
- Encapsulation and isolation of environments
- Give the user control/choices
- Multiple window managers - Cinnamon/Mate/GnomeShell
- Helpldesk role in Linux support, management
- Flexible to meet customer needs/desires
- Software versioning (Matlab 2014/2015/2016/…)
- Mainstream stable OS selection
  - Software supports multiple common OS “branches”
  - RedHat based (rpm)
  - Debian based (pkg)
  - Other?
- Handle kernel upgrades
- Distro agnostic
- Flexible in SW and config methods applied
- Because they want Ubuntu
- Low touch baseline

Flexible, Modular, Highly Available

- Options for local replication in case of network failure
- High availability & no single point of failure
- Handle disconnected systems
- Keep module like system

Environment Modules – Previous Iteration

- Package management independent of OS
- Supports multiple versions of software
- Simple selection of available software
- Hassle-free Environment (shell, variables, etc)
- Modules written in TCL
- No mechanism for tracking module usage

LMOD Environmental Modules

- Developed at Texas Advanced Computing Center
- Modules written in Lua
  - Handles legacy TCL files
- Supports hierarchical modules
- Supports module usage tracking
  - Lmod module hook -> rsyslog to db server -> ingest syslog into local mysql db (or influx)
- https://www.tacc.utexas.edu/research-development/tacc-projects/lmod

Software Modules in ELI

- Only building x86_64 software in new environment
- package-version naming convention
  - ex. /software/python-2.7.13
- Automount /software (local and remote).
- Environment setup under any OS
- Default software selection

Software, Class, and Environment Modules

$ module avail

```
------------------------------------------------ /etc/modulefiles/env -------------------------------------------------
                                                     ...........
----------------------------------------------- /etc/modulefiles/class ------------------------------------------------
                                                     ...........
---------------------------------------------- /etc/modulefiles/software ----------------------------------------------
   CDFPlayer/10.4.0                  ctos/13.20.200                    llvm/3.5CS225                python/3.4.3      (D)
   Cabal/1.24.2.0                    cuda-toolkit/4.0                  llvm/3.7.1            (D)    python3/3.4
   OOF2/2.1.12                       cuda-toolkit/4.1          (D)     m4/1.4.17b                   python3/3.4.1
   PETSc/3.5.1                       cuda-toolkit/5.0                  mathematica/8.0              python3/3.5.2     (D)
   QtSpim/9.1.7                      cuda-toolkit/6.5                  mathematica/9.0              qhull/2012.1
   SciPy-Stack/2.7.10-x86_64         cuda-toolkit/8.0                  mathematica/10.0      (D)    qwt/5.2.2
   Synopsys_x86-64/2015              customic/06.16.030        (D)     mathematica/10.2             racket/6.1.1
   abaqus/6.10-1             (D)     customic/51.41.151                matlab/R2009a                root/5.30.00
   abaqus/6.11-1                     cx/I-2013.12                      matlab/R2009b                root/5.32.03      (D)
   abaqus/6.13-2                     dc/G-2012.06-SP5-5                matlab/R2010b                root/6.02.08
   abaqus/6.14-1                     diffpy/1.0                        matlab/R2011a                root/6.06.02
   abaqus-research/6.10-1    (D)     disper/0.3.0                      matlab/R2013a                ruby/2.3.0
   abaqus-research/6.11-1            dorsal/1.0.0                      matlab/R2013b                sage/6.5
                                                     ...........
```

$ module load matlab/R2015a

Software Module - TCL

```
#%Module1.0####################################################################
##
##
## null modulefile
##
## modulefiles/null.  Generated from null.in by configure.
##

#@name Python 3.4.3
#@description Python is an interpreted, interactive, object-oriented programming language
#@website http://python.org/

proc ModulesHelp { } {
        global version

        puts stderr "\tThis module sets up the environment for Python 3.4.3"
}


module-whatis   "setup Python 3.4.3"

eval set  [ array get env SOFTPATH ]
eval set  [ array get env DISTARCH ]

set modulename [string map {/ -} [module-info name]]
set appdir     $SOFTPATH/$modulename

prepend-path   PATH            $appdir/bin
```

Software Module - Lua

```
/etc/modulefiles/software/python3/3.5.2.lua
  -------------------------------------------
  help([[
  For detailed instructions, go to:
  http://python.org
  ]])

whatis("Version: 3.5.2")
  whatis("Keywords: python, python3")
  whatis("URL: http://python.org")
  whatis("Description: python3")
  prepend_path( "PATH", "/software/python-3.5.2-cent7/bin")
  prepend_path( "LD_LIBRARY_PATH","/software/python-3.5.2-
  cent7/lib")
```

Software Usage

```
# ./analyzeLmodDB --sqlPattern '%matlab%' counts
Module path                                        Distinct Users
-----------                                        --------------
/etc/modulefiles/software/matlab/R2015a                      3453
/etc/modulefiles/software/matlab/R2011a                       593
/etc/modulefiles/software/matlab/R2014b                        82
/etc/modulefiles/software/matlab/R2014a                        51
/etc/modulefiles/software/matlab-research/R2016a               50
…
```

```
# ./analyzeLmodDB --sqlPattern '%%' counts
Module path                                        Distinct Users
-----------                                        --------------
/etc/modulefiles/software/matlab/R2015a                      3453
/etc/modulefiles/software/python3/3.4.1                      1468
/etc/modulefiles/software/python3/3.5.2.lua                  1177
/etc/modulefiles/software/lc3tools/12                        1128
/etc/modulefiles/software/intel-license/ews                   990
…
```

```
# ./analyzeLmodDB --start '2017-01-17' --end '2017-05-12' \
                  --sqlPattern 'ccoughle' modules_used_by
Module path                                              User Name
-----------                                              ---------
/etc/modulefiles/class/ece483.lua                         ccoughle
/etc/modulefiles/software/abaqus/6.10-1                   ccoughle
/etc/modulefiles/software/altera/13.1                     ccoughle
/etc/modulefiles/software/anaconda/2.2.0                  ccoughle
/etc/modulefiles/software/cadence/Aug2016.lua             ccoughle
…
```

Authn/Authz

Authentication and Authorization

- Authentication
  - Kerberos vs the AD
- Authorization:
  - Home Brewed solution
    - Most managed machines
    - Crawl AD and build flat files on ~30 min intervals
  - SSSD LDAP
    - Used in the CBTF and CS VM Farm
    - Three different versions: CBTF, Ubuntu, Centos

Home Brewed Solution

- Python script crawls and flattens our AD structure
- Resulting files pushed to Git repo
- Salt runs create local accounts on machines
  - Make file to create db for users, groups, etc under /var
  - Exists in parallel to system accounts under /etc
- Advantages:
  - Stable and has been in use for years
  - We can fake GID, Shell and HomeDir (not in AD)
- Disadvantages:
  - Very much a home brewed solution

SSSD LDAP

- Point SSSD to ldap.ad.uillinois.edu
- Configured in salt
- Advantages:
  - Portable and supported product
  - Quicker turn around for group and user changes
- Disadvantages:
  - No GID, Shell, HomeDir for users in AD
  - Groups searches can cause issues
  - Failure to pass group membership to Pam on login (sometimes)

How we have used SSSD

- CBTF:
  - No Authorization based on groups. Everyone can log into the machines
  - Authorization is really physical space control
- CSVM FARM:
  - Create groups for each set of machines
  - Restricted group/user search base
  - Override GID
  - Used /etc/security/access to do 1-1 student to VM mappings

Going Forward with SSSD

- Need to fix the missing AD fields
- Figure out the issues with PAM and login
- Groups search is better now (Thanks Frank Penrose)

Problems

Things we are still working on

File Sharing:

- What We Wanted:
  - User level authorization of file permissions
  - Machine Authorization via netgroups
  - Native cloud synchronization clients
- What We Got:
  - Machine netgroup auth
  - Cloud storage access via web browser
  - Software Distribution via nfs share

Desktop Managers

- Gnome 3 The epic pile of cross dependency hell
  - Terrible compatibility with FastX (Graphical remote desktop)
  - Gnome Keyring prompt causes tidal wave of support tickets

Ubuntu

- Privacy concerns about Amazon integration with Unity
  - Switched to Mate, done!
- Fail2ban single jail takes precedence over other configured jails
  - Delete /etc/fail2ban/jail.d/defaults-debian.conf
- Removal of Popcorn and Popularity Contest
- Postfix is installed and listening on all interfaces by default
  - Postfix shouldn't be running on the desktops, removed!
- Stop Ubuntu from nagging to upgrade to new release
  - Edit /etc/update-manager/release-upgrades

NVidia video drivers

- **Previous version**: Custom built init.d script and hand-maintained nVidia repo of nvidia.run packages.
  - Port Attempt # 1: Systemd makes this difficult due to needing to complete before graphical interface comes up.
  - Salt-State Attempt: Only worked because lab environment has standard hardware, and required ~3 reboots.
- **Current Solution**: Local mirror of select packages From Elrepo, such as nVidia drivers

Questions?